Socket and host configuration

This set of directives allows you to configure your virtual hosts. In practice, this materializes by creating `server` blocks that you identify either by a hostname or by an IP address and port combination. In addition, some directives let you fine-tune your network settings by configuring the TCP socket options.

listen

Context: `server`

Specifies the IP address and/or the port to be used by the listening socket that serves the website. Sites are generally served on port `80` (the default value) via HTTP, or `443` via HTTPS.

Syntax: `listen [address][:port] [additional options];`

Additional options:

- `default_server`: Specifies this `server` block to be used as the default website for any request received at the specified IP address and port
- `ssl`: Specifies that the website should be served over SSL
- `spdy`: Enables support for the SPDY protocol if the SPDY module is present
- `proxy_protocol`: Enables the PROXY protocol for all the connections accepted on this port
- Other options are related to the `bind` and `listen` system calls: `backlog=num`, `rcvbuf=size`, `sndbuf=size`, `accept_filter=filter`, `deferred`, `setfib=number`, and `bind`

Examples:

```
listen 192.168.1.1:80;
listen 127.0.0.1;
listen 80 default;
listen [:::a8c9:1234]:80; # IPv6 addresses must be put between square
brackets
listen 443 ssl;
```

This directive also allows Unix sockets:

```
listen unix:/tmp/nginx.sock;
```

server_name

Context: `server`

This assigns one or more hostnames to the `server` block. When Nginx receives an HTTP request, it matches the `Host` header of the request against all the `server` blocks. The first `server` block to match this hostname is selected.

Plan B: if no `server` block matches the desired host, Nginx selects the first `server` block that matches the parameters of the `listen` directive (such as `listen *:80` would be a catch-all for all the requests received on port `80`), giving priority to the first block that has the `default` option enabled on the `listen` directive.

Note that this directive accepts wildcards as well as regular expressions (in which case, the hostname should start with the ~ character).

Syntax: `server_name hostname1 [hostname2…];`

Examples:

```
server_name www.website.com;
server_name www.website.com website.com;
server_name *.website.com;
server_name .website.com; # combines both *.website.com and website.
com
server_name *.website.*;
server_name ~^(www)\.example\.com$; # $1 = www
```

Note that you may use an empty string as the directive value in order to catch all the requests that do not come with a `Host` header, but only after at least one regular name (or "_" for a dummy hostname):

```
server_name website.com "";
server_name _ "";
```

server_name_in_redirect

Context: `http`, `server`, `location`

This directive applies to the case of internal redirects (for more information about internal redirects, check the *Rewrite Module* section given further on in this chapter). If set to `on`, Nginx uses the first hostname specified in the `server_name` directive. If set to `off`, Nginx uses the value of the `Host` header from the HTTP request.

Syntax: `on` or `off`

Default value: `off`

server_names_hash_max_size

Context: `http`

Nginx uses hash tables for various data collections in order to speed up the processing of requests. This directive defines the maximum size of the server names hash table. The default value fits with most configurations. If this needs to be changed, Nginx automatically tells you so on startup, or when you reload its configuration.

Syntax: Numeric value

Default value: `512`

server_names_hash_bucket_size

Context: `http`

Sets the bucket size for server names hash tables. You should only change this value if Nginx tells you to.

Syntax: Numeric value

Default value: `32` (or `64`, or `128`, depending on your processor cache specifications)

port_in_redirect

Context: `http`, `server`, `location`

In case of a redirect, this directive defines whether or not Nginx should append the port number to the redirection URL.

Syntax: `on` or `off`

Default value: `on`

tcp_nodelay

Context: `http`, `server`, `location`

Enables or disables the `TCP_NODELAY` socket option for keep-alive connections only. Quoting the Linux documentation on sockets programming:

> "*TCP_NODELAY is for a specific purpose; to disable the Nagle buffering algorithm. It should only be set for applications that send frequent small bursts of information without getting an immediate response, where timely delivery of data is required (the canonical example is mouse movements).*"

Syntax: `on` or `off`

Default value: `on`

tcp_nopush

Context: `http`, `server`, `location`

Enables or disables the `TCP_NOPUSH` (FreeBSD) or `TCP_CORK` (Linux) socket option. Note that this option applies only if the `sendfile` directive is enabled. If `tcp_nopush` is set to on, Nginx attempts to transmit all the HTTP response headers in a single TCP packet.

Syntax: `on` or `off`

Default value: `off`

sendfile

Context: `http`, `server`, `location`

If this directive is enabled, Nginx uses the `sendfile` kernel call to handle file transmission. If disabled, Nginx handles the file transfer by itself. Depending on the physical location of the file being transmitted (such as NFS), this option may affect the server performance.

Syntax: `on` or `off`

Default value: `off`

sendfile_max_chunk

Context: `http`, `server`

This directive defines the maximum size of data to be used for each call to `sendfile` (read the previous section).

Syntax: Numeric value (size)

Default value: `0`

send_lowat

Context: `http`, `server`

This is an option that allows you to make use of the `SO_SNDLOWAT` flag for TCP sockets under FreeBSD only. This value defines the minimum number of bytes in the buffer for output operations.

Syntax: Numeric value (size)

Default value: `0`

reset_timedout_connection

Context: `http`, `server`, `location`

When a client connection times out, its associated information may remain in memory depending on its state. Enabling this directive will erase all memory associated with the connection after it times out.

Syntax: `on` or `off`

Default value: `off`

Paths and documents

This section describes the directives that configure the documents that should be served for each website, such as the document root, the site index, error pages, and so on.

root

Context: `http`, `server`, `location`, `if`. Variables are accepted.

This directive defines the document root containing the files that you wish to serve to your visitors.

Syntax: Directory path

Default value: `html`

```
root /home/website.com/public_html;
```